Software Technology to Enable Reliable High-Performance Distributed Disk Arrays

Second Year Progress Report

Michael S. Warren, Ryan Joseph, John Wofford

Mail Stop B227
Theoretical Astrophysics
Los Alamos National Laboratory
Los Alamos, NM 87545

LA-UR 07-5798

Abstract

It is currently possible to construct a single-node RAID storage system with a 14 Terabyte capacity using commodity serial-ATA hard disk drives for less than $10,000. A cluster of such systems (a distributed disk array) would be able to provide over a petabyte of storage for less than 1 million dollars. Obtaining reliablity and good performance from such a system is the focus of our project.

Over the course of this project, we have established a number of testbed and production systems containing over 200 Terabytes of storage. Significant progress to date includes detailed failure statistics on a variety of disk drives, performance benchmarks on a number of different systems, modifications to the Linux ATA-over-Ethernet (aoe) driver to reliably support RAID-5 and RAID-6 arrays across multiple cluster nodes, and development of the InfiniBlok Linux device driver to support block-level I/O over Infiniband devices. A system developed in the course of this project has been named as a finalist for the SC07 storage challenge.

1 Introduction

Persistent data storage is a fundamental prerequisite for all information science and technology projects. The advent of commodity microprocessors with adequate floating-point performance and low-priced fast ethernet switches contributed to the emergence of Beowulf clusters in the mid-90s, which revolutionized parallel computing at the departmental scale [1, 2, 3]. We are currently in the midst of a similar revolution in scalable data storage, due to the dramatic decline in the price of commodity disk drives and 10-gigabit networking technologies.

Vast amounts of relatively inexpensive storage offer significant opportunities to develop new approaches to scientific problems, most of which will be difficult to predict in advance. The cost per Gbyte for SATA disk storage is currently about $0.35 for 1 Terabyte drives. We have demonstrated single-node fault-tolerant 14 Terabyte servers for a total cost of under $10k (see Table 1). Used in a parallel cluster environment, a Petabyte disk array with achievable read/write bandwidth that greatly exceeds available Gigabit local and wide-area networking technology is possible.

There are current commercial products which provide highly-reliable and high-performance disk storage (Network Appliance and Panasas, for example). However, these products cost about a 5-10x premium to the equivalent storage using commodity-off-the-shelf components (which is perhaps not coincidentally the typical factor between the cost of a Beowulf system and a commercial supercomputer). Additional costs are accrued over the life of the commercial product due to their proprietary nature and maintenance contracts. Additionally, the greater CPU/storage ratio in a DDA offers techniques which are not possible in traditional RAID arrays.

As larger and more precise surveys of the Universe are performed, connecting the observational data with a consistent theoretical understanding of the distribution and properties of galaxies, stars and other objects becomes more difficult. During the next decade large projects will generate more than 100 times as much observational data as has been gathered in all of our prior history. A precise understanding of the systematic errors in both the observations and our models is critical to the determination of the fundamental parameters of the Universe.

We have already amassed an on-line collection of a number of large astronomical datasets (DSS digitized Schmidt plates, SDSS imaging and spectroscopy, FIRST radio survey, USNO-B1.0 object catalog, 2MASS, etc.) with the intention of collecting and organizing the world's astronomical data and making it universally accessible and useful. These observational datasets, along with simulation data, currently comprise over 100 Terabytes of information. This astronomical data repository provides a unique resource for testing the data storage systems which are the focus of this project.

In the same way in which special-purpose telescopes are now required to obtain the best catalogs of objects in the sky, a focused effort involving state-of-the-art parallel computer hardware and software combined with theoretical and observational expertise is required in order to model the Universe which led to this observed distribution of stars, dust and galaxies.

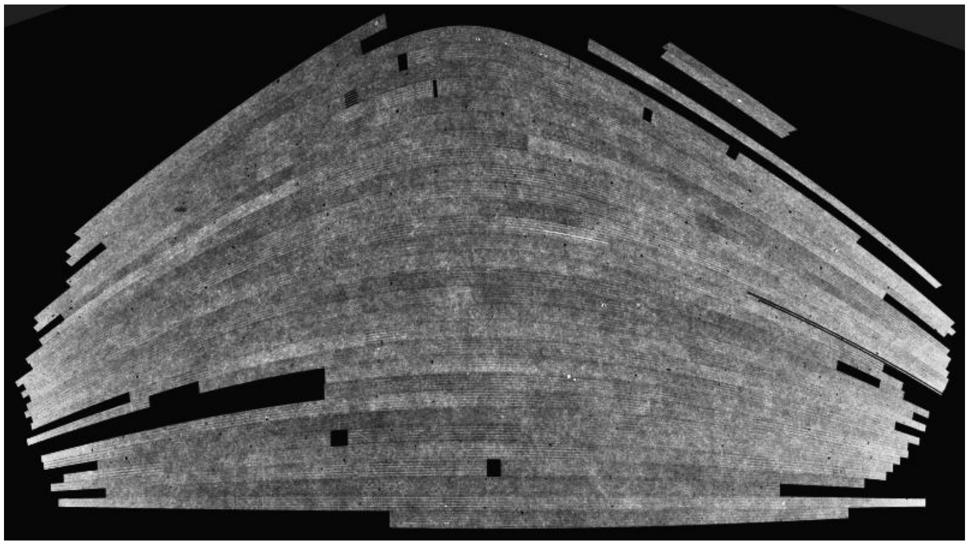

Figure 1: A figure showing the source density in the Sloan Digital Sky Survey Data Release 5 Object Catalog. Data analysis for this figure was performed at LANL on a prototype of the system described here. The geometry is an equal-area HEALpix decomposition of the celestial sphere. The pixel intensity is proportional to the log of the number objects in the pixel. The dark bands are gaps in the survey, most of which have been filled in in the latest data release (DR6). The raw data consists of object catalogs which contain over 200 million objects in over 300,000 individual files. Each object comprises 2732 bytes of data, for a total object catalog data volume of about 0.5 Tbyte in this region of the sky.

2 Progress During Year 2

2.1 Reference "I/O Brick" design

We investigated a number of storage systems in order to determine the optimal "building block" for larger systems. Our current reference design is listed in Table 1, which has a storage capacity of 14 Tbytes at a cost of about $10000. We currently have 8 such systems, which will be the testbed hardware for further software development. I/O read bandwidths for each node are about 750 Mbytes/sec, while writes are about 350 Mbytes/sec with the 3ware card and 630 Mbytes/sec with the Areca card (see Table 3).

2.2 Hardware Monitoring

Recent work on over 100,000 disk drives [4] have gathered failure statistics on commodity disk drives under real-world conditions, and demonstrate that the failure rate of the more reliable recent models is about 3% per year. These statistics are consistent with our own (relatively) limited experience. While higher than the MTBF claims of manufacturers, such a failure rate is well within the ability of a part-time system administrator to manage (i.e. a petabyte array with 1000 drives would suffer a disk failure about every 12 days).

Qty.	Price	Ext.	Description
2	290	580	AMD Opteron dual-core 2214 2.2GHz
16	355	5680	Hitachi 1TB SATA 7200RPM
1	45	45	PCI Video Card
1	895	895	3ware 9650SX-16 RAID 6 card
1	475	475	TYAN S2915A2NRF 4x PCIe
4	95	380	Kingston 1GB DDR2-667 ECC/REG
1	25	25	Addonics IDE to CF adapter
1	42	42	4GB Compact Flash (system disk)
1	795	795	Myricom Myri-10g network card
1	1159	1159	RM3U316 16x SATA chassis
1	72	72	Assembly
Total		$10148	$724 per Tbyte

Table 1: 3U Rackmount storage system pricing, July 2007

2.3 ATA over Ethernet

This project depends fundamentally on scaling disk storage from a single system to a parallel system. The networking abstraction used to communicate between systems is an important facet of this approach.

The iSCSI specification was developed in order to standardize communication with network attached SCSI devices. In the first year of our summer intern, Ryan Joseph, investigated all of the major iSCSI implementations to see if they would provide a solid foundation to proceed from. The conclusions from this research were that iSCSI was over-engineered, and no robust implementations currently exist.

Our initial approach used the "Network Block Device" (a standard part of the Linux kernel) which provided the functionality required for network attached storage. This approach was documented in our first year progress report. After some modifications to the driver to make it more robust, we were able to implement a RAID system using nbd over multiple storage nodes. After some experience with this system, the nbd approach was abandoned for the more mature and reliable ATA over Ethernet (aoe).

AoE is used to achieve a very basic level RPC mechanism between a client and an ATA device server (www.coraid.com/documents/AoEr10.txt). We have demonstrated the ability to create multiple-node disk arrays using the Linux aoetools. The log below shows a 24 Tbyte software RAID0 filesystem created out of three storage nodes.

```
# mdadm --create /dev/md0 --chunk=128 --level=0 --raid-devices=3 \
        /dev/sda1 /dev/etherd/e0.0 /dev/etherd/e0.1
mdadm: array /dev/md0 started.

# cat /proc/mdstat
```

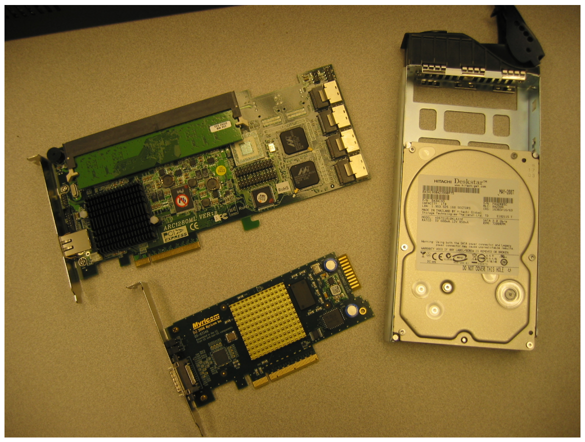

Figure 2: On the left, seven 20-packs of Hitachi 1TB drives (140 Terabytes) ready to be unpacked and installed. On the right, enabling technologies: an Areca 16-port RAID card, a Myricom 10-gigabit network card, and a Hitachi 1 Terabyte hard drive.

```
Personalities : [raid0]
md0 : active raid0 etherd/e0.1[2] etherd/e0.0[1] sda1[0]
      18754129536 blocks 128k chunks

# mkfs.xfs -f /dev/md0; mount /dev/md0 /raid

# df -h /raid
Filesystem            Size  Used Avail Use% Mounted on
/dev/md0               24T  528K   24T   1% /raid
```

2.4 InfiniBlok

InfiniBlok is a light-weight kernel module developed by John Wofford at LANL that allows a remote storage device to appear as if it were present in a local machine. It exploits all of the advantages of InfiniBand to give low latency and low CPU overhead. The goal is to create a protocol similar to ATA over Ethernet (AoE) or Network Block Devices (NBD) that will make use of InfiniBand RDMA and reliable connections. Like NBD, InfiniBlok speaks the language of the Linux kernel directly; it translates kernel operations into packets, sends them across the InfiniBand

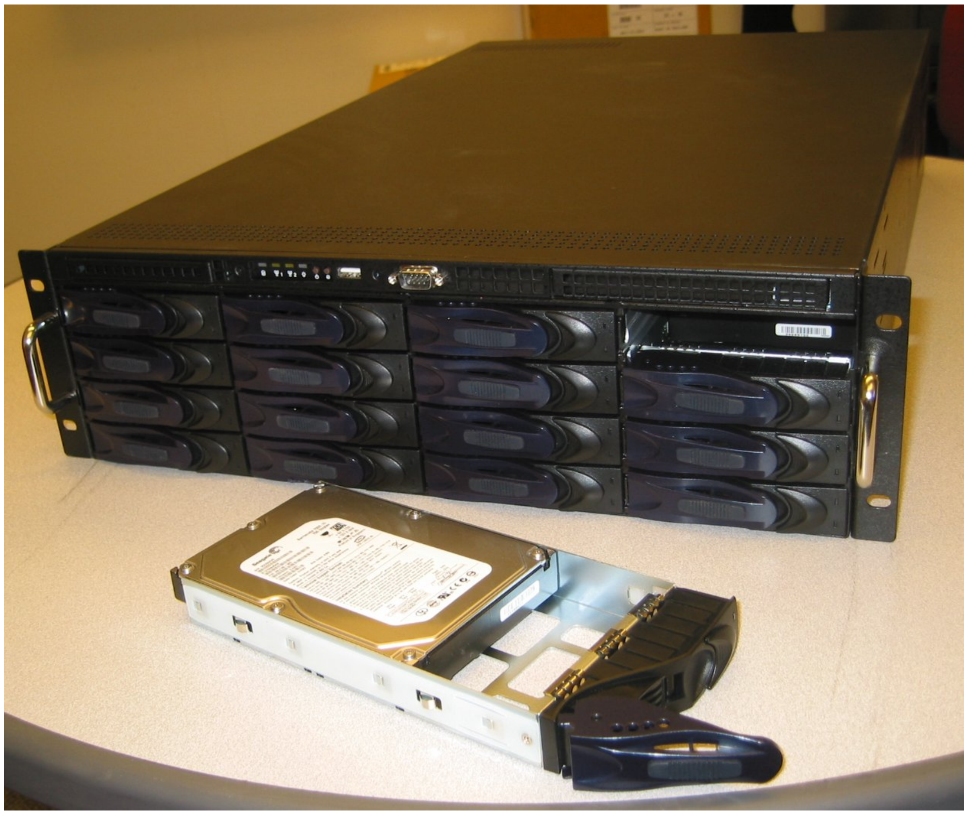

Figure 3: A 3U 16-disk RAID storage node. A Petabyte storage system using technology currently available would consist of 72 such nodes, and would cost less than $1M.

link, and translates them back into kernel operations. All data transfers use RDMA. Unlike either AoE or NBD, InfiniBlok happens in the kernel for both client and server connections. In fact, both client and server end are handled by the same Linux kernel module.

The InfiniBlok protocol is very simple. It consists of three packets: *geometry* (GEO), *make requests* (MRQ) and *completion* (CMP). When a device is served InfiniBlok attaches to the block device, giving InfiniBlok direct access to send and receive commands to and from the physical storage device driver. InfiniBlok then listens for connections. After the block device is claimed by the server, a client machine can connect, giving it access to that block device as if it were local to the client machine.

When an InfiniBand reliable connection is established between a server (containing storage) and a client (connecting to storage) the server sends a GEO packet describing the block device to be served, including the device's capacity, hardware block size and CHS geometry (so that the GETGEO ioctl can be supported). Once the client receives the GEO packet it creates a block device on the client end that mimics the block device on the server end.

When the client's block device receives a read or write request the kernel passes the InfiniBlok driver a *bio* structure containing buffers that are either to be filled with data from the storage device (read request) or be transfered to the storage device (write request). InfiniBlok maps all of the *bio* buffers for RDMA transfer and translates all relevant information from the *bio* into an MRQ packet which it sends to the server.

When the server receives the MRQ it acts differently depending on whether the request is a read request or a write request. If the request is a read request InfiniBlok creates a *bio* structure on the server end with empty buffers and passes that *bio* to the driver for the physical storage device. The storage device driver will then fill the empty buffers with data from the disk. Once the storage device driver has processed the submitted *bio* it notifies InfiniBlok of the completion status including possible errors and how many bytes were transfered. Next, InfiniBlok RDMA writes all of the data from the server side *bio* into the client side *bio*. When these RDMA transfers are complete a CMP packet is sent from the server to the client indicating all of the information that was returned by the actual storage driver (error status, number of bytes transfered).

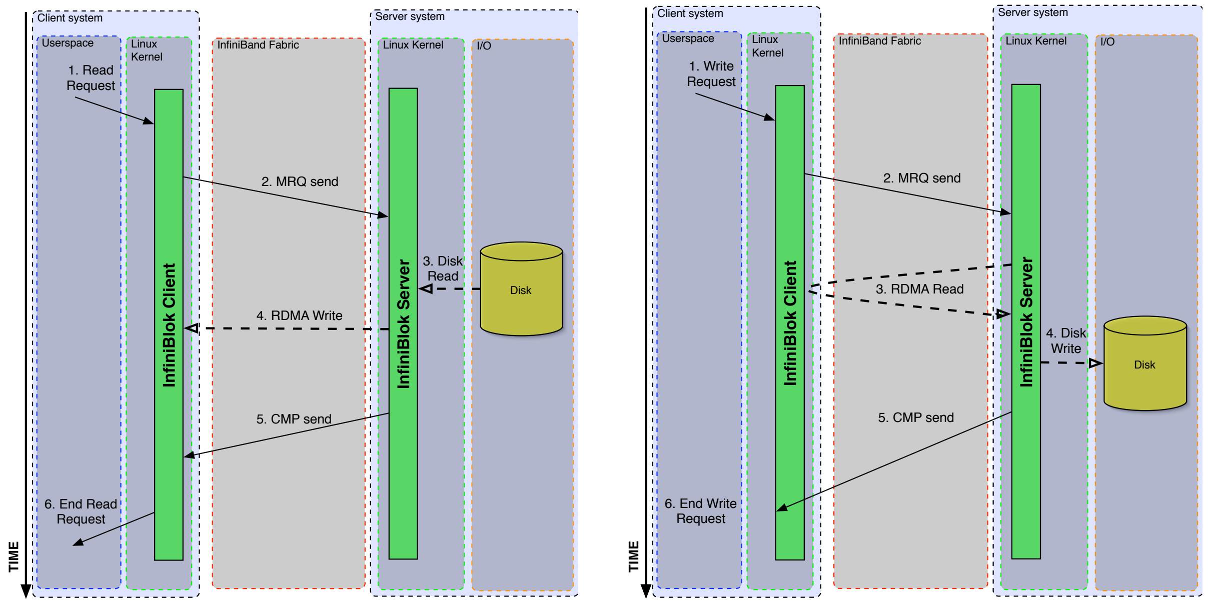

Figure 4: InfiniBlok protocol

If the request is a write request similar processes happen but in a slightly different order. When the server receives the MRQ it begins RDMA read transfers from the client *bio* into a server *bio*. Once the transfers are complete the server side *bio*, which now has buffers full of information to be written to disk, is submitted to the storage device driver. When the storage device driver finishes processing the *bio* a CMP packet is sent immediately to the client containing the same information as for a read request.

When the client receives a CMP packet it then returns that same information to the kernel regarding the original request, signaling the completion (or error) of the requested operation. The InfiniBlok driver can handle many such requests concurrently. The number of simultaneous client and server connections is limited only by memory (system and InfiniBand), and the driver is designed to make full use of multiprocessor systems.

As a result of speaking the language of the kernel directly, InfiniBlok creates a block device on the client end which is in every way indistinguishable from a local device. The InfiniBlok device can be partitioned, formatted, mounted or even used as a target for LVM or software RAID. This latter use is of particular interest. LVM or software RAID can be used to combine a number of InfiniBlok devices into a single high speed, low latency block device which requires little CPU overhead.

InfiniBlok has distinct advantages over its Ethernet based competitors, AoE and NBD, in its ability to use the RDMA and built–in reliable connection features of InfiniBand. Further, since packets do not need to carry the data that is being transferred, large bundles of RDMA transfers can be sent in a single MRQ, allowing fewer packets to be transmitted. Even as Ethernet speeds increase and latencies decrease, these advantages should keep InfiniBlok competitive.

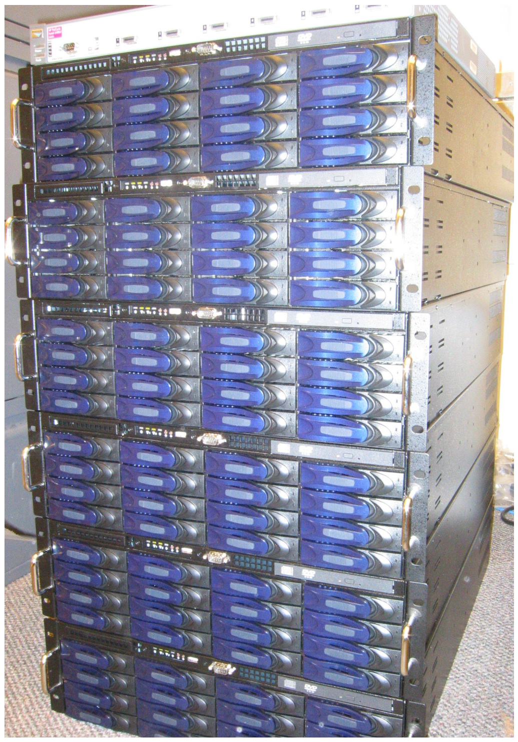
Figure 5: Six 3U 16-disk RAID storage nodes, with a raw storage capacity of 96 Tbytes. On top of the storage nodes is an HP 6-port 10-GigE Procurve switch. Configured as a RAID6 x RAID5 disk array, the aggregate storage of the system is 70 Tbytes.

The major alternatives which use InfiniBand include NFS over RDMA, SCSI RDMA Protocol (SRP) and iSER. NFS over RDMA carries with it the large protocol overhead of NFS and does not allow the creation of a block device on the local machine, making it a fundamentally different type of storage solution. SRP and iSER are similar protocols in that they both communicate by passing SCSI commands. SRP ties into the SCSI layer, while supplies a set of extensions for the use of iSCSI. Both protocols provide a robust set of features, and both use RDMA to transfer data. There are two main differences between InfiniBlok and SRP or iSCSI. The first difference is that InfiniBlok ties into the simple command set of the Linux block layer while SRP and iSCSI use the SCSI stack. The second difference is that InfiniBlok's protocol is much lighter weight than SRP and iSCSI. InfiniBlok assumes a trusted network and a minimal set of features requiring no login and only basic read and write functionality. In the spirit of keeping it simple, InfiniBlok may provide a straightforward and reliable open-source alternative to the more complex existing solutions.

2.5 Benchmarks

We evaluated a number of systems in order to quantify the performance of single disks, as well as different types of hardware and software RAID5/RAID6 systems.

	Write	Re-Write	Read	Re-Read
3ware Hardware RAID-0	27.44	20.09	62.73	61.24
3ware Hardware RAID-5	27.44	20.09	62.73	61.24
3ware Software RAID-5	95.99	81.28	230.18	226.63
NBD baseline, 1kb BS	20.76	19.14	78.63	79.19
14-disk NBD RAID-5	91.23	93.46	56.38	56.78
tmpfs RAID-0	107.96	125.31	95.78	96.77
tmpfs RAID-5	81.07	87.52	90.86	90.90
3ware Software RAID-0	282.06	160.19	389.53	390.14
3ware Hardware RAID-5	161.26	98.77	254.21	257.66
3ware Software RAID-6	81.37	73.36	176.58	174.71
Highpoint Software RAID-0	353.18	247.63	443.32	445.43
Highpoint Software RAID-5	147.90	92.31	207.90	208.21
Apple Xserve RAID-5	43.02	40.73	133.77	147.33

Table 2: Hardware Storage Benchmarks (all data in Mbytes/sec). These data were gathered from 2003-2006, and show performance of the previous generations of PCI-X based hardware, and the Linux network block device across Gigabit Ethernet.

2.6 Single Node Performance

The maximum sustained read rates from the current generation of SATA disk drives now exceed 75 Mbytes/sec.

```
# hdparm -tT /dev/sdb

/dev/sdb:
 Timing cached reads:   1956 MB in  2.00 seconds = 978.72 MB/sec
 Timing buffered disk reads:  226 MB in  3.00 seconds =  75.31 MB/sec
```

	Write	Re-Write	Read	Re-Read
3ware 9650SE-16ML RAID6	313.3	381.0	763.6	763.3
Areca ARC-1261 RAID6	635.5	665.3	738.2	737.9

Table 3: Current hardware Storage Benchmarks (all data in Mbytes/sec). File size used was 1 Tbyte. These data were gathered on the prototype system described in Table 1 in July 2007, and show performance of the current generation of PCI Express based hardware using 1 Terabyte Hitachi SATA drives.

2.7 Network Performance

Netperf results between two storage nodes with Myricom 10-gigE hardware was measured at 8893 Megabits/sec using the `TCP SENDFILE TEST`. The myri10ge driver was recompiled from the sources distributed from Myricom, and optimized according to the instruction posted at their web site.

	Write	Re-Write	Read	Re-Read
NFS	178.9	187.3	136.6	133.6
AoE	253.0	253.2	134.7	135.0
InfiniBlok	432.7	323.2	184.7	185.6

Table 4: Networked storage performance between two storage nodes (Mbytes/sec) using iozone. NFS used default settings for Fedora Linux version 7. ATA over ethernet used the default version shipped with FC7. NFS and AoE results used a Myri-10g card. Infiniblok used Mellanox Technologies MT25208 InfiniHost III cards. Read rates are a factor of 4-5 below the potential performance, indicating substantial opportunity for optimization to improve the results.

3 Plans for Year 3

In the short term, we will be producing a report to submit to the SC07 storage challenge in November 2007 (see `sc07.supercomputing.org/schedule/event_detail.php?evid=11239`) to showcase our approach in a friendly competition [5]. Our approach to the final year of this project will be driven primarily by application needs in the area of astronomy and astrophysics. In particular, optimizing the analysis of large astronomical image datasets, and computing statistics from large N-body simulations have already pointed us at the most critical areas to optimize. The two main areas to improve are performance and reliability.

Much progress has been made with Linux kernel developers addressing "deadlock" problems common to network storage applications. Several interesting subtleties exist in the interaction between the block layer and the network layer in the Linux kernel. In particular, when the system becomes short on memory, dirty pages must be written to their backing store. However, if the act of writing those pages requires a memory allocation (as it normally would when writing to a TCP socket) the system will deadlock. A good summary of the problem and solution is available at `http://lkml.org/lkml/2006/8/8/349`.

The most significant performance bottleneck in the analysis of astronomical datasets appears to analogous to filesystem fragmentation. The outstanding performance of current RAID systems depends on having large contiguous blocks of data on the storage system, so that parallel reads and disk read-ahead can maximize the utilization of the I/O system. Unless extreme care is taken in the transfer of data onto the system (which requires knowledge of the order in which files will be analyzed) the files tend to become randomly distributed on the device, which causes the read-ahead to fetch a large amount of data which is not necessary. The simplest solution is to encourage the use of large files (Gbytes, rather than Mbytes). Alternatively, for support of legacy datasets like SDSS, it should be possible to write software analogous to the “defragmenter” necessary for reasonable performance with primitive filesystems to re-arrange files into the optimal order. In our case, however, it is not fragmentation within a file which is the problem, but disorder in the overall organization of files being fed to the analysis pipeline.

Acknowledgements

We wish to thank Wu-Chun Feng and M. Patrick Goda for their significant contributions to earlier work on this project. This work was carried out under the auspices of the National Nuclear Security Administration of the U. S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396.

References

[1] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, pages 11–14, 1995.

[2] M. S. Warren, D. J. Becker, M. P. Goda, J. K. Salmon, and T. Sterling. Parallel supercomputing with commodity components. In H. R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA’97)*, pages 1372–1381, 1997.

[3] M. S. Warren, J. K. Salmon, D. J. Becker, M. P. Goda, T. Sterling, and G. S. Winckelmans. Pentium Pro inside: I. A treecode at 430 Gigaflops on ASCI Red, II. Price/performance of $50/Mflop on Loki and Hyglac. In *Supercomputing ’97*, Los Alamitos, 1997. IEEE Comp. Soc.

[4] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST07)*, 2007.

[5] M. S. Warren and J. Wofford. Astronomical data analysis with commodity components. Finalist for the SC07 Storage Challenge, Reno NV, Nov. 10-16, 2007.